Discrete Applied Mathematics 157 (2009) 2958–2963

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Note

A note on the minimum bounded edge-partition of a tree

Shane Dye *

Department of Management, University of Canterbury, Private Bag 4800, Christchurch, New Zealand

ARTICLE INFO

Article history:
Received 19 December 2007
Received in revised form 17 March 2009
Accepted 1 April 2009
Available online 29 April 2009

Keywords:
Tree
Partition
Optimization problem
Approximation algorithm

ABSTRACT

Minimum bounded edge-partition divides the edge set of a tree into the minimum number of disjoint connected components given a maximum weight for any component. It is an adaptation of the uniform edge-partition of a tree. An optimization algorithm is developed for this NP-hard problem, based on repeated bin packing of inter-related instances. The algorithm has linear running time for the class of 'balanced trees' common for the stochastic programming application which motivated investigation of this problem.

Fast 2-approximation algorithms are formed for general instances by replacing the optimal bin packing with almost any bin packing heuristic. The asymptotic worst-case ratio of these approximation algorithms is never better than the absolute worst-case ratio of the bin packing heuristic used.

1. Introduction

Given a tree, T, edge weights, w, and a weight bound, K, the minimum bounded edge-partition is a partition of the edge set into the minimum number of connected components, with component weights no more than K. This short note presents an optimization algorithm for this problem and an adaptation which produces a family of 2-approximation algorithms.

The problem is a variation of that of the *uniform* edge-partition of a tree studied in Wu et al. [10], which fixes the number of components, k, in the partition and optimizes the spread of component weights. Specifically, the objective is to minimize the ratio of the maximum to the minimum component weight. For a tree with n edges, they show that a solution with a ratio of at most 3 can be found in $O(n \log k)$ time. The bound is improved to a ratio of at most 2 for $k = 2, 3, 4$. The reader is referred to the paper for further details.

Variants of edge-partitioning in trees have also been studied as a travelling salesmen problem on a tree [1,8].

The variation studied in this note arises in the context of a stochastic programming algorithm [5]. The algorithm decomposes the stochastic program into subproblems corresponding to subtrees of the scenario tree. The subtrees form an edge-partition with the weight of a subtree indicating subproblem size. Decompositions with few larger subproblems generally perform better than those with many smaller subproblems so long as the subproblem size is below certain limits based on the optimization software and machine used.

To formally present the problem, let $E(T)$ denote the edges of tree T and $w(T)$ the sum of its edge weights. The edge with end-points u and v is denoted as uv or vu; its weight is w_{uv}. Given maximum subtree weight $K > 0$, a K-pack of T with size m is a collection $\{T_1, \ldots, T_m\}$, where (1) each T_i is a connected subgraph of T, (2) the T_i are pairwise edge-disjoint, (3) the union of all subgraphs forms T, and (4) $w(T_i) \leq K$, for each T_i. An implied condition of the definition is that edges may not be split over multiple T_i.

Minimum bounded edge-partition (MBP)

Instance: Tree T, weight bound $K > 0$ and edge weights w in $[0, K]$.
Goal: Find a K-pack of T of minimum size.

* Tel.: +64 3 364 2886; fax: +64 3 364 2020.
E-mail address: shane.dye@canterbury.ac.nz.

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.04.001

This problem is NP-hard. The reduction from the three-partition problem used in Wu et al. [10] applies to the problem studied here.

The remainder of the paper is structured as follows. An optimization algorithm based on optimal bin packing is described in Section 2. It also describes a class of trees for which the algorithm has linear running time. For other trees, 2-approximation algorithms are formed by replacing the optimal bin packing with various bin packing heuristics. Worst-case performance is investigated in Section 3. Concluding remarks are given in Section 4.

2. An optimization algorithm

A K-pack is formed using bin packing to combine sets of trial trees connected to the same vertex. The trial trees have a special structure, with a root attached to a single edge. These edges index the trial trees, as the set $\mathcal{E}$.

The algorithm starts with no trial trees and all leaf vertices forming a set of active vertices. Active vertices are roots of trial trees which have all but one connected edge in $\mathcal{E}$. Each active vertex, u, is considered in turn. Edge set $\mathcal{E}_u$ indexes the trial trees for which u is the root. If the tree formed by combining these trial trees with the remaining edge of u does not violate the weight bound, this combined tree becomes a new trial tree, replacing all of those indexed by $\mathcal{E}_u$. On the other hand, if the combined tree violates the weight bound, bin packing is used to optimally combine the trial trees into components of the partition being formed. One of these components could become part of a larger component, and is combined with the remaining edge uv to form a trial tree. Choosing the smallest possible component ensures the best K-pack for the remaining tree. To ensure this, the bin packing used is modified to minimize both the number of bins and the size of the smallest bin. The vertex v becomes active if all but one of its connected edges are in $\mathcal{E}$. Vertex label $d(v)$ is used to keep track of the number of connected edges not in $\mathcal{E}$. The edge label $\ell(uv)$ keeps track of the weight of the trial tree indexed by uv.

In the formal description the trial trees are denoted as $T'(uv)$, $\mathcal{P}$ is the K-pack being constructed and A the set of currently active vertices. Set operations applied to trees form new trees, in the obvious way. Edges are used as trees where appropriate.

Minimum K-Pack Algorithm

(1) For instance (T, w, K), set $\mathcal{P} = \emptyset$, $\mathcal{E} = \emptyset$ and make A the set of leaf vertices. For each vertex v set $d(v)$ as its degree.
(2) Choose $u \in A$ and remove it. Put $\mathcal{E}_u = \{ u'u \in \mathcal{E} \}$. If $d(u) = 0$ jump to Step (5); otherwise let uv be the remaining edge incident to u but not in $\mathcal{E}_u$.
(3) Put $\mathcal{E} = (\mathcal{E} \setminus \mathcal{E}_u) \cup \{uv\}$, set $d(v) = d(v) - 1$ and if $d(v) = 1$ add v to A.
(4) If $w_{uv} + \sum_{u'u \in \mathcal{E}_u} \ell(u'u) > K$, go to Step (5); otherwise no new trees are created. Put $T'(uv) = uv \cup \bigcup_{u'u \in \mathcal{E}_u} T'(u'u)$, and $\ell(uv) = w_{uv} + \sum_{u'u \in \mathcal{E}_u} \ell(u'u)$. Return to Step (2).
(5) Solve a modified bin packing instance with bins of capacity K and one item for each edge in $\mathcal{E}_u$ using edge labels as item sizes. The objective is to first minimize the number of bins, m, then (for this number of bins) minimize the weight of bin 1.
   (a) For bins $b = 1, \ldots, m$, construct tree T_b by combining the trial trees indexed by the edges associated with items packed into bin b.
   (b) If $d(u) = 0$ or $w(T_1) + w_{uv} > K$ put $f = 1$; otherwise put $f = 2$. For bins $b = f, \ldots, m$, add T_b to $\mathcal{P}$. If $d(u) = 0$, A is empty and the algorithm is complete.
   (c) If $f = 2$, put $T'(uv) = uv \cup T_1$ and $\ell(uv) = w(T_1) + w_{uv}$. Otherwise, put $T'(uv) = uv$ and $\ell(uv) = w_{uv}$. In both cases return to Step (2).

The overall running time is $O(n + nB(d, K))$, where $n = |E(T)|$ and d is the maximum degree of any vertex. $B(d, K)$ is the worst-case running time for the bin packing with d items and capacity K; it is not polynomially bounded unless P = NP. Optimization algorithms based on branch-and-bound have been developed for solving the original bin packing problem, for example [7,9]. These can be modified for the required objective by using the single objective function $\min(K + 1)m + s_1$ where m is the number of bins used and s_1 the weight of bin 1. The coefficient of m ensures that minimizing bins is the first priority since $s_1 \leq K$. The changes required could potentially reduce the efficiency of these approaches and further study is needed to determine the full implications.

In general the running time is dominated by the term $nB(d, K)$. When d is bounded the algorithm has a theoretical running time of $O(n)$.

The following results are needed to show that the algorithm solves the problem posed. They provide insight into the structure of an optimal solution.

Given instance (T, w, K), the following notation is used. $\mathcal{M}(T, w, K)$ is the collection of all minimum bounded edge-partitions. For edge uv let $T_{uv}(u)$ be the component of T containing u after uv is removed. For K-pack $\mathcal{Q}$, $\{\mathcal{Q}_{uv}(u), \mathcal{Q}_{uv}(v), \{Q_{uv}\}\}$ partitions $\mathcal{Q}$ into trees wholly contained within $T_{uv}(u)$ or $T_{uv}(v)$, with Q_{uv} the tree containing uv. Put $T'_{uv}(u) = T_{uv}(u) \cup uv$ and $Q'_{uv}(u) = Q_{uv}(u) \cup \{Q_{uv}\}$. For $\mathcal{C}$, an arbitrary collection of subtrees of T, define $G[\mathcal{C}]$ to be the graph induced by $\mathcal{C}$.

The following properties are easy to verify.

Lemma 1. *For the Minimum K-Pack Algorithm the following hold.*

(1) *The algorithm generates a K-pack.*
(2) *Each vertex takes the role of u in Step* (2) *exactly once.*

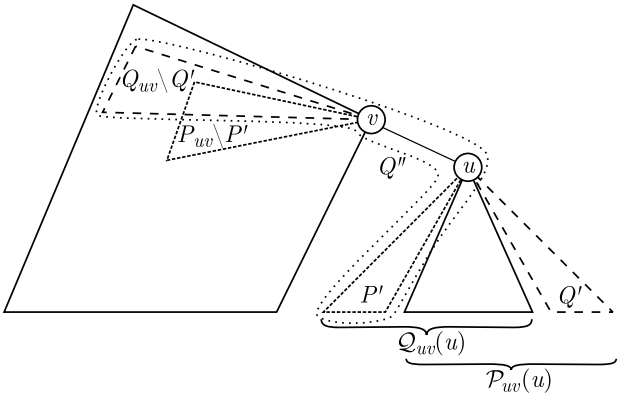

Fig. 1. Various subtrees used to construct the required K-pack, $Q'' = (Q_{uv} \setminus Q') \cup P'$.

(3) *For $v \in A$, all incident edges, except at most one, will be in $\mathcal{E}$.*
(4) *For any bin of any bin packing instance (including a bin 1), all edges of all trial trees associated with that bin will be in the same $T_i \in \mathcal{P}$.*

The next result shows circumstances where decomposition may be used.

Lemma 2. *Let $\mathcal{P} \in \mathcal{M}(T, w, K)$ and $\{\mathcal{P}_1, \mathcal{P}_2\}$ partition $\mathcal{P}$ such that $G[\mathcal{P}_i]$ is connected for $i = 1, 2$. Then $\mathcal{P}_i \in \mathcal{M}(G[\mathcal{P}_i], w, K)$ for $i = 1, 2$ and if $\mathcal{Q}_i \in \mathcal{M}(G[\mathcal{P}_i], w, K)$, $i = 1, 2$, then $\mathcal{Q}_1 \cup \mathcal{Q}_2 \in \mathcal{M}(T, w, K)$.*

Proof. It is easy to check that $\mathcal{P}_i$ is a K-pack of $(G[\mathcal{P}_i], w, K)$, $i = 1, 2$, and that $\mathcal{Q}_1 \cup \mathcal{Q}_2$ is a K-pack of (T, w, K). Therefore,

$$|\mathcal{Q}_1 \cup \mathcal{Q}_2| = |\mathcal{Q}_1| + |\mathcal{Q}_2| \leq |\mathcal{P}_1| + |\mathcal{P}_2| = |\mathcal{P}| \leq |\mathcal{Q}_1 \cup \mathcal{Q}_2|,$$

and $|\mathcal{Q}_i| = |\mathcal{P}_1| + |\mathcal{P}_2| - |\mathcal{Q}_{3-i}| \geq |\mathcal{P}_i| \geq |\mathcal{Q}_i|$. □

The following lemma shows the connection between the bin packing and minimal bounded edge-partitions

Lemma 3. *For any instance (T, w, K), let u be the first vertex for which Step (5) is applied. There exists $\mathcal{Q} \in \mathcal{M}(T, w, K)$ with the following property. If uv is defined in Step (2), $\mathcal{Q}_{uv}(u)$ corresponds to a feasible solution of the bin packing associated with u. Otherwise, $\mathcal{Q}$ has this property and the bin packing solution is optimal.*

Proof. For the existence of $\mathcal{Q}$, the same argument works for both cases, so assume uv was defined. It is easy to verify that if each tree of $\mathcal{Q}_{uv}(u)$ contains at least one edge connected to u, this corresponds to a feasible bin packing. Assume that $\mathcal{Q}$ is chosen so that $\mathcal{Q}_{uv}(u)$ contains the minimal number of non-conforming trees. This means that there is some $T' \in \mathcal{Q}_{uv}(u)$ containing an edge uy where T' does not fully contain $T'_{uy}(y)$. Vertex y was previously active, so $w(T'_{uy}(y)) \leq K$. Replacing all trees in $\mathcal{Q}$ covering $T'_{uy}(y)$ with $\{T'_{uy}(y), T' \setminus T'_{uy}(y)\}$ gives a K-pack with no more trees and fewer non-conforming trees, a contradiction.

For the case where optimality is asserted, bin packing is applied to the whole tree. The optimal bin packing corresponds to a K-pack with the same number of trees as bins used; a better bin packing contradicts optimality of $\mathcal{Q}$. □

Theorem 4. *The Minimum K-Pack Algorithm solves MBP.*

Proof. The algorithm finishes after examining each vertex once. Induction is used on the number of edges. For instances with one edge the algorithm obviously solves MBP. Assume that the algorithm solves MBP for any instance with n edges or fewer and let (T, w, K) be any instance with $n + 1$ edges. Let $\mathcal{P}$ be the K-pack produced by the algorithm, u be the first vertex for which Step (5) is applied and m the number of bins used. Let $\mathcal{P}_B \subseteq \mathcal{P}$ be the m trees corresponding to the bins packed in this first Step (5), including bin 1 whose tree may have been generated at some later iteration. If $\mathcal{P}_B = \mathcal{P}$ then $\mathcal{P}$ is optimal by Lemma 3.

Otherwise, choose $\mathcal{Q} \in \mathcal{M}(T, w, K)$ satisfying the conditions of Lemma 3 such that $Q_{uv} \cap T_{uv}(u)$ has the least weight.

If $Q_{uv} \cap T_{uv}(u) = \{u\}$, using $G[\mathcal{P}_{uv}(v)] \subset G[\mathcal{Q}'_{uv}(v)]$, the induction hypothesis and Lemmas 2 and 3, $|\mathcal{Q}_{uv}(u)| = m$ and $|\mathcal{P}_{uv}(v)| \leq |\mathcal{Q}_{uv}(v)| + 1$. This gives $|\mathcal{P}| = |\mathcal{P}_{uv}(v)| + m \leq |\mathcal{Q}_{uv}(v)| + 1 + |\mathcal{Q}_{uv}(u)| = |\mathcal{Q}|$.

If $Q_{uv} \cap T_{uv}(u) \neq \{u\}$, let $Q' = Q_{uv} \cap T_{uv}(u)$ and $P' = P_{uv} \cap T_{uv}(u)$. By choice of $\mathcal{Q}$ the collection $\{Q'\} \cup \mathcal{Q}_{uv}(u)$ corresponds to a feasible bin packing and either (i) $w(Q') < w(P')$ or (ii) $w(Q') \geq w(P')$. For case (i), by the optimality criteria of the bin packing, $1 + |\mathcal{Q}_{uv}(u)| > m$ and reasoning similar to that for the previous case gives the result. For case (ii), an optimal K-pack is created which decomposes in a way that allows the induction hypothesis to apply. Fig. 1 illustrates the various subtrees involved. Let $Q'' = (Q_{uv} \setminus Q') \cup P'$. Using Lemma 2 and the induction hypothesis, it is straightforward to check that $\mathcal{Q}' = \mathcal{Q}_{uv}(v) \cup \{Q''\} \cup \mathcal{P}_{uv}(u)$ is an optimal K-pack with $|\mathcal{Q}| = |\mathcal{Q}'| = |\mathcal{P}|$.

Invoking induction gives the result. □

The above shows that it is possible to form a minimum bounded edge-partition recursively, using bin packing to optimally pack trial trees connected to the same vertex.

Trees for which the vertex-degree is constant across each stage are called "balanced trees" in the stochastic programming literature [4]. Balanced scenario trees arise as a result of modelling or scenario tree generation. The Minimum K-Pack Algorithm has a linear running time for such trees when the edge weights satisfy an additional requirement which is reasonable in the context of balanced scenario trees.

Lemma 5. *Let (T, w, K) be an instance of MBP for which T is a balanced tree with root r and for which all edges at the same stage have the same weight. The Minimum K-Pack Algorithm runs in $O(n)$ time if r is the last active vertex chosen.*

Proof. $|A| \geq 2$ for all but the final iteration, so it is possible to choose r last. The edge labels for all edges at the same stage will be equal after they have each been active. This means that for each bin packing instance from Step (5) all items will have the same size. Such instances can be optimally packed in time linear in the degree of vertex u using a next-fit approach. The sum of degrees in a tree is $O(n)$, so the Minimum K-Pack Algorithm will have running time $O(n)$. □

3. Approximation algorithms

For instances not meeting the criteria of Lemma 5, approximation algorithms are investigated. A class of approximation algorithms for MBP evolve in a straightforward manner from the optimization algorithm by replacing the optimal bin packing in Step (5) with a bin packing heuristic, choosing bin 1 to be the least filled bin. The approximation algorithm is called a BPA if the bin packing heuristic satisfies the following (weak) performance property.

Property 1. *All bins packed by the bin packing heuristic, except at most one bin, have total weight of more than one half of the bin capacity.*

Many common bin packing heuristics have this property, for instance, first fit and best fit (see, for example, [3]), and the H_7 linear time heuristic of Békési and Galambos [2]. Any heuristic which does not have this property can be extended by a linear time procedure (combining pairs of non-compliant bins) to produce an improved solution which does satisfy the property. The reader is directed to the review by Coffman et al. [3] for further information on bin packing heuristics.

The following results show that BPA's provide a 2-approximation for MBP. In what follows, let $m^P(T, w, K)$ be the number of trees in a minimum bounded edge-partition and $m^H(T, w, K)$ be the number of trees in K-pack constructed by a BPA. The first result provides a lower bound for MBP.

Lemma 6. *For instance (T, w, K), $m^P(T, w, K) \geq \lceil w(T)/K \rceil$.*

Proof. Let $\mathcal{P} = \{T_1, \ldots, T_m\} \in \mathcal{M}(T, w, K)$. Since $\mathcal{P}$ forms a partition of $E(T)$,

$$w(T)/K = \frac{1}{K} \sum_{i=1}^{m} w(T_i) \leq m.$$

The result follows since m is an integer. □

The following result is used to bound the number of trees from above. For edge set E, $w(E)$ is the sum of its edge weights.

Lemma 7. *For instance (T, w, K) let $\{E_1, \ldots, E_m\}$ be a set packing of $E(T)$ with $w(E_i) \geq K$ for $i = 1, \ldots, m$ and $w(E_k) > K$ for at least one k; then $m \leq \lceil w(T)/K \rceil - 1$.*

Proof. Since $\{E_1, \ldots, E_m\}$ is a set packing of $E(T)$, the E_i are disjoint and their union is a subset of $E(T)$, so,

$$w(T)/K = w(E(T))/K \geq \frac{1}{K} \sum_{i=1}^{m} w(E_i) > m.$$

The results follows as m is integer. □

The following theorem shows that BPA's are 2-approximations.

Theorem 8. *For instance (T, w, K) any BPA has an absolute worst-case approximation ratio of 2. In addition, when $w(T) > 0$, $m^H(T, w, K) \leq 2\lceil w(T)/K \rceil$.*

Proof. If $w(T) \leq K$ the result trivially holds. Assuming $w(T) > K$, an edge-partition satisfying Lemma 7 is constructed. Let $\mathcal{P}$ be the K-pack produced by the BPA. By Property 1 and the criteria for adding T_b to $\mathcal{P}$, all but at most one $T_i \in \mathcal{P}$ will have either (i) $w(T_i) > \frac{1}{2}K$ or (ii) $w(T_i) \leq \frac{1}{2}K$ and there is uv such that $w(T_i) + w_{uv} > K$. For case (ii), label the tree holding the associated uv as $t(T_i)$. These $t(T_i)$ are distinct and satisfy case (i) since $w(T_i) \leq \frac{1}{2}K$ implies that $w_{uv} > \frac{1}{2}K$.

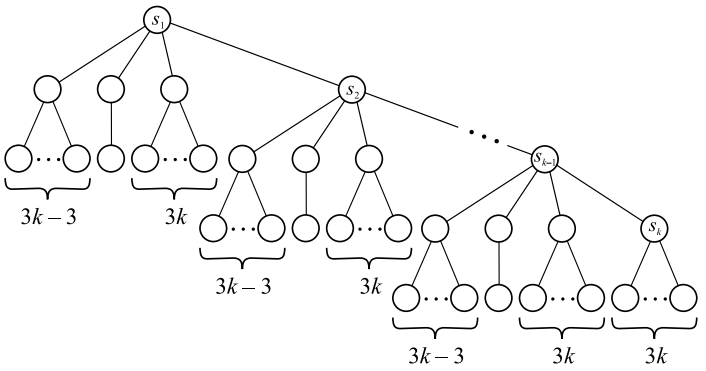

Fig. 2. Worst-case first-fit BPA instance (unit edge weights, $K = 6k - 1$).

Create a collection of edge sets, $\mathcal{D}$, by combining edges from each case (ii) T_i with $t(T_i)$ and then combining edges from arbitrary pairs of the remaining case (i) trees. Now, $\mathcal{D}$ satisfies the requirements of Lemma 7 and $|\mathcal{D}| \geq \frac{1}{2}m^H(T, w, K) - 1$. Applying Lemmas 6 and 7 gives

$$m^H(T, w, K) \leq 2\lceil w(T)/K \rceil \leq 2m^P(T, w, K).$$

The result follows. □

For some BPA the constant approximation ratio is tight. The set of examples below show this for the first-fit BPA. The first-fit bin packing heuristic has an absolute worst-case approximation ratio of less than or equal to 1.75 [3].

For any integer $k \geq 2$, recursively define a tree as follows. Each step, t, uses special vertex s_t. Begin with $t = 1$ and create vertex s_1. Add four child vertices to s_t in order: c_1, c_2, c_3 and c_4, with the following child vertices (each of which will be a leaf vertex). For c_1, create $3k - 3$ child vertices, for c_2 create 1 child vertex and for c_3 create $3k$ child vertices. Vertex c_4 becomes the next special vertex, s_{t+1}. If $t < k - 1$, increment t by 1 and repeat the creation of child vertices. Otherwise, create $3k$ child vertices for s_k; the tree is complete. Fig. 2 illustrates the tree. All edges have unit weight and $K = 6k - 1$.

Applying the first-fit BPA to this instance, the leaf vertices, all c_1–c_3 vertices and s_k can become active and chosen as u without using Step (5). At this point the remaining vertices are s_1 to s_{k-1}; these can be chosen as u in reverse order. The first such iteration has $u = s_{k-1}$ and the bin packing instance has item sizes $3k-2, 2, 3k+1$ and $3k+1$. First fit creates three bins, the smallest with weight $3k$. Two trees are added to $\mathcal{P}$, leaving $\ell(s_{k-2}s_{k-1}) = 3k + 1$. The remaining iterations follow the same pattern, with $u = s_t$, bin packing item sizes $3k - 2, 2, 3k + 1$ and $3k + 1$, two trees added to $\mathcal{P}$ and $\ell(s_{t-1}s_t) = 3k + 1$. The final iteration uses $u = s_1$ and the same bin packing instance but adds three trees to $\mathcal{P}$. A total of $2(k - 1) + 1 = 2k - 1$ trees are in $\mathcal{P}$.

Applying the optimization, vertices may be chosen in the same order as for the BPA, above. The iterations are the same until $u = s_{k-1}$, where, as before, the bin packing has item sizes $3k - 2, 2, 3k + 1$ and $3k + 1$. The optimal bin packing uses two bins, the smallest holding weight $3k + 3$. One tree is added to $\mathcal{P}$ and $\ell(s_{k-2}s_{k-1}) = 3k + 4$. For the remaining iterations, when $u = s_t$, the bin packing has item sizes $3k - 2, 2, 3k + 1$ and $6k - 3t - 2$, the optimal packing uses two bins, one tree is added to $\mathcal{P}$, and $\ell(s_{t-1}s_t) = 6k - 3t + 1$. The final iteration uses $u = s_1$, has item sizes $3k - 2, 2, 3k + 1$ and $6k - 5$, the optimal packing uses two bins, and two trees are added to $\mathcal{P}$ giving a total of k trees.

4. Concluding remarks

For any BPA it is straightforward to generate examples with an arbitrarily large number of edges corresponding to a bin packing instance which gives the *absolute* worst-case ratio for the bin packing heuristic used. This means that the *asymptotic* worst-case performance of any BPA cannot be better than the bin packing heuristic's *absolute* worst-case ratio. The previous example shows that the asymptotic ratio of the BPA can be bigger than this.

Corresponding tight examples are not known for the BPA using first-fit decreasing, best-fit decreasing (see for example [3]), the H_4 linear time heuristic of Martel [6] or the H_7 linear time heuristic of Békési and Galambos [2]. For these BPA's the worst-case examples found have ratios of 1.5. These examples correspond to absolute worst-case examples for the bin packing heuristic. Future work could be directed at closing the gap between the bound and the worst-case examples known.

Acknowledgements

The author is grateful for the excellent comments from the two referees. Their suggestions have helped to clarify and improve the paper.

References

[1] I. Averbakh, O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, Discrete Applied Mathematics 75 (3) (1997) 201–216.
[2] J. Békési, G. Galambos, A 5/4 linear time bin packing algorithm, Journal of Computer and System Sciences 60 (2000) 145–160.
[3] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: A survey, in: D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing, Boston, 1996, pp. 46–93.
[4] J. Dupačová, G. Consigli, S.W. Wallace, Scenarios for multistage stochastic programs, Annals of Operations Research 100 (2000) 25–53.
[5] S. Dye, Subtree decomposition for multistage stochastic programs, Working paper, Stochastic Programming e-Print Series, http://www.speps.info/ (2003).
[6] C.U. Martel, A linear time bin-packing algorithm, Operations Research Letters 4 (4) (1985) 189–192.
[7] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & sons, 1990.
[8] H. Nagamochi, K. Okada, A faster 2-approximation algorithm for the minmax p-travelling salesmen problem on a tree, Discrete Applied Mathematics 140 (2004) 103–114.
[9] J.M. Valério de Carvalho, Exact solution of bin-packing problems using column generation and branch-and-bound, Annals of Operations Research 86 (1999) 629–659.
[10] B.Y. Wu, H.-L. Wang, S.T. Kuan, K.-M. Chao, On the uniform edge-partition of a tree, Discrete Applied Mathematics 155 (2007) 1213–1223.